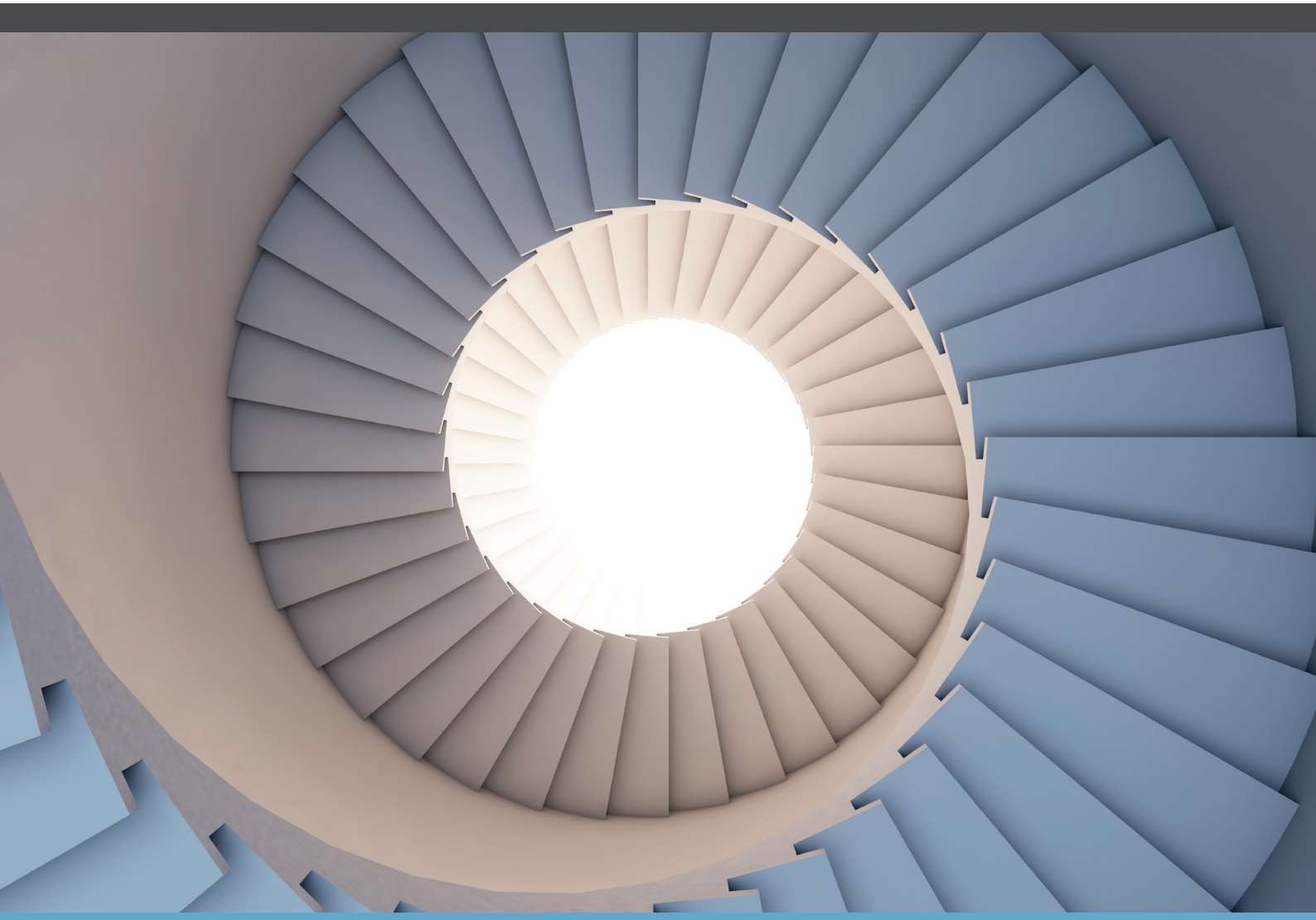
2-2016

Research Software Sustainability: Report on a Knowledge Exchange Workshop

Simon Hettrick
The Software Sustainability Institute

Knowledge Exchange



Research Software Sustainability

Report on a Knowledge Exchange Workshop

Author
Simon Hettrick, The Software Sustainability Institute

February 2016

DFG Jisc CSC DEff SURF

“Research Software Sustainability:
Report on a Knowledge Exchange Workshop”

Author
Simon Hettrick,
The Software Sustainability Institute

Contents

An Introduction to Knowledge Exchange

The Knowledge Exchange (KE) partners are five key national organisations within Europe tasked with developing infrastructure and services to enable the use of digital technologies to improve higher education and research: CSC in Finland, DEFF in Denmark, DFG in Germany, Jisc in the UK and SURF in the Netherlands.

Our five partners share a clear vision that scholarship should be open. Through Knowledge Exchange we are working together to support the development of digital infrastructure to enable open scholarship. We are raising our collective voice to inform national and international policies and promote common approaches, so that it becomes easier for scholarship to cross national boundaries.

We share our knowledge, experiences and resources.

Our vision

Digital technologies create innovative opportunities to advance research and higher education.

Open scholarship is one of these opportunities. Opening up access to scientific outputs (including publications, data and software) and encouraging open methods and collaboration can:

- improve transparency and engender greater trust in research
- increase the effective sharing and use of research
- support wider participation in research

All of these possibilities are underpinned by digital technologies.

Our vision is to enable open scholarship by supporting an information infrastructure on an international level.

Our mission

Knowledge Exchange activities support the individual agendas of the five partner organisations. They also advance progress towards achieving our shared vision.

We are increasing the impact of partner activities by exchanging knowledge between experts in the area of digital technologies for research and higher education.

We are building on those exchanges to inform developments in information infrastructure, including technical as well as the organisational, policy and economic aspects.

We are exchanging best practice, practical solutions and innovative approaches to improve all aspects of each partner's performance. This will ensure that they can create more effective solutions.

Contact
Web: knowledge-exchange.info
Email: office@knowledge-exchange.info
Twitter: @knowexchange
Tel: +44 203 697 5804

1 Executive summary

Without software, modern research would not be possible. Understandably, people tend to marvel at results rather than the tools used in their discovery, which means the fundamental role of software in research has been largely overlooked. But whether it is widely recognised or not, research is inexorably connected to the software that is used to generate results, and if we continue to overlook software we put at risk the reliability and reproducibility of the research itself.

The adoption of software is accompanied by new risks - many of which are unknown to the majority of the research community. The practices of software sustainability minimise these risks and help to increase trust in research results, increase the rate of discovery, increase return on investment and ensure that research data remains readable and usable for longer.

Funders are well aligned with the goals of software sustainability: both seek to support reliable, trusted research. This means that funders are well placed to play a pivotal role in advocating software sustainability. Funders can help raise awareness, and can make some simple, low-cost changes, such as encouraging the adoption of software management plans, that could lead to significant improvements in the software used in research.

Improving software sustainability requires a number of changes: some technical and others societal, some small and others significant. We must start by raising awareness of researchers' reliance on software. This goal will become easier if we recognise the valuable contribution that software makes to research - and reward those people who invest their time into developing reliable and reproducible software. We must educate the research community on the issues raised by software adoption, and provide training in the software engineering skills that are needed to overcome them. We cannot rely on researchers to adopt all of the skills needed for software sustainability, we must also allow research groups to recruit software experts, and we must create organisations that develop and disseminate expertise in software sustainability.

The adoption of software has led to significant advances in research. But if we do not change our research practices, the continued rise in software use will be accompanied by a rise in retractions. Ultimately, anyone who is concerned about the reliability and reproducibility of research should be concerned about software sustainability. This argument alone may rely too much on the stick and not enough on the carrot. To that end, we must also show that software sustainability promises to identify and make available the software that is most likely to advance research.

2 About the workshop

The Knowledge Exchange Workshop on Research Software Sustainability took place on 1-2 October 2016 at the DFG Offices in Berlin.

The meeting was organised and chaired by Bas Cordewener, Matthias Katerbow and Sarah James. Many thanks are due to those who contributed their time and expertise to this report. In particular to all those who presented and contributed at the workshop:

Patrick Aerts, Hans Bennis, Timo Borst, Daan Broeder, Christopher Brown, Neil Chue Hong, Bas Cordewener, Mustafa Dogan, Peter Doorn, Matthew Dovey, Martin Hammitzsch, Simon Hettrick, Sarah James, Matthias Katerbow, Brian Matthews, Cedric Nugteren, Andreas Raabe, Arfon Smith, Stefan Strathman, Rob van Nieuwpoort, Stefan Winkler-Nees and Andreas Zeller.

This report is based on discussions that took place during the workshop. It was written by Simon Hettrick and edited by Bas Cordewener, Matthias Katerbow and the workshop participants.

Footnotes

1 "Creative Commons — Attribution 4.0 International — CC BY ..." 2013. 21 Dec. 2015 (https://creativecommons.org/licenses/by/4.0).

3 Background

To understand the importance of software sustainability, one must understand some key facts about software and its role in research.

3.1 Definitions

Three terms are often used, almost interchangeably, when describing approaches to extending software lifetime: sustainability, preservation and archiving. There is no fixed definition for these terms, so for the purposes of this report the following definitions will be used.

Research software (as opposed to simply **software**) is software that is developed within academia and used for the purposes of research: to generate, process and analyse results. This includes a broad range of software, from highly developed packages with significant user bases[2,3,4] to short (tens of lines of code) programs written by researchers for their own use.

Software sustainability describes the practices, both technical and non-technical, that allow software to continue to operate as expected in the future. A constant level of effort is required to maintain the software's operation.

Software preservation is an approach to extend the lifetime of software that is no longer actively maintained. There are different approaches, as described below, which vary in the effort required and the likelihood of success.

Software archiving is one important aspect of software preservation. It is the process of storing a copy of software so that it may be referred to in the future. For example, for the purposes of reproducibility, to store a copy of the exact software that was used to generate a result in a publication.

3.2 Introduction to software sustainability

Software is used at every stage of research, and across all disciplines, to generate, process and analyse results. Yet it is a relatively new addition to the researcher's toolbox, especially when we move from the **traditional** computational fields and into new disciplines where software is being rapidly adopted. Consequently, software best practice and the skills needed to apply it are not yet embedded in the research community. Software promises to open new vistas of research, but it is also associated with new risks. To mitigate these risks, we must adopt the practices of software sustainability.

Software sustainability[5] is vital to research because it is inexorably linked to the reliability and reproducibility of results. Although these concepts are well understood in research, they tend to be overlooked when it comes to research software. Software is a tool like any other, so it must stand up to the same scrutiny. Improving the reliability and reproducibility of software promises benefits that will be felt across the research community, but it requires change across that same community: from funders, research organisations, publishers and researchers.

Footnotes

2 "BALL Project: About." 2005. 21 Dec. 2015 (**ball-project.org**).

3 "dmacrys - UCL Chemistry." 2008. 21 Dec. 2015 (**chem.ucl.ac.uk/basictechorg/dmacrys/index.html**).

4 "Praat - Phonetic Sciences, Amsterdam." 2003. 21 Dec. 2015 (**fon.hum.uva.nl/praat**).

5 See **Appendix 1**, presentation 1.

Software sustainability defies easy definition. In general, it can be seen as the practices that allow software to continue to function as expected in the future, but this definition makes sustainability sound straightforward. It is not.

Software sustainability is an umbrella term that covers a wide array of issues that are technical, cultural and political in nature. There is no one-size-fits-all solution: sustainability is highly specific to the software under consideration. Technology can help in some areas, but it cannot solve the problem alone. Instead, we must look to change community practices to overcome societal barriers that prevent or inhibit sustainability. Some of these changes will be simple and easy to adopt, others will challenge long-standing academic practices, meaning that time and incentives will be needed to persuade the community to change.

Software has a lifecycle: it is conceived, matures and decays. At any point during this lifetime, an intervention may be required to keep the software viable and to help extract the greatest value from the investment made into it. It is not feasible or desirable to sustain all software. Inevitably some will outlive its usefulness, and some will become unsustainable due to forces beyond the control of the software's owners or user community. Rather than sustain all software, advocates of software sustainability argue that resources should be concentrated on sustaining software that is most useful. At the current time, this is a difficult and occasionally arbitrary process because there is no agreed system for measuring the contribution of software to research. We must develop such a system. If software can achieve credit, then software developers can measure their contribution to research in much the same way that publications are currently used by researchers. By rewarding time invested into software development (by both researchers and software developers) we will incentivise the development of higher quality research software across a wider range of areas.

There is cause for hope to be found in the world of research data management. The campaign for understanding the importance of data to research is at least 10-15 years ahead of the campaign for software. Although there is still room for improvement in this area, the campaign has succeeded in getting data onto the research agenda and has motivated change at every level of the community. The momentum behind the campaign for data can be used to raise awareness of software. After all, data is nothing without the software needed to read, analyse and visualise it. At the same time, we must ensure that stakeholders understand that software is not data: we can leverage the lessons learned by that campaign, but we cannot simply re-apply to software the best practices for research data management.

Software sustainability will require changes across the research community, some of which are non-trivial - but this can be an iterative journey. The small and easy changes can showcase benefits that entice people to tackle the larger challenges.

3.3 Software is not data

It is interesting to note the parallels between data and software. Over the last 10-15 years there has been a concerted effort to raise awareness of the importance of data to research. Interest in Research Data Management (RDM), and investment into understanding and then disseminating best practices have considerably improved how data is handled and preserved. **But software is not data**. Whereas data is largely static, software is a living entity that must evolve and adapt to the constant changes in its environment. If it does not evolve, software will decay. It can be argued that data is subject to decay, but this is not the case. It is actually the software used to store and read the data that decays.

Software is always reliant on other software for its correct operation. The arrangement can be thought of as a pyramid or stack: with the software the user wants to use at the top, which is dependent on a second tier of other software, which may itself be dependent on third tier of other software, and so on. The **other software** is known as the **dependent software** and it includes everything from the operating system, system libraries, and other necessary packages (e.g. a browser, JRE, etc.). Software decay[6] occurs because a change to any of the dependent software can affect the operation of the software higher up the stack, and the risk of this occurring is often high because there is usually a large number of software dependencies.

If we attempt to preserve software, it quickly becomes out of step with its dependent software. Within a short space of time these changes accumulate to the point

where the software can no longer function. If software is to continue to be of use, it must be sustained rather than preserved.

3.4 Software lifecycle

Software should be seen as a living entity that may pass through a number of stages during its lifetime. At any stage, and for many different reasons, the software may come to the end of its lifetime.

Research software can be developed by a single researcher or software developer, or by a team. A significant amount of research software is developed by a single researcher (often a postdoctoral student or postgraduate) to solve a problem specific to their research. If the software proves useful, it can attract users. Growth generally begins within the research group, then within the general field in which the software was developed and then, less frequently, the software may achieve large-scale adoption across a number of domains. It is entirely possible that code developed by a postdoctoral student may become a highly successful software project with users from around the globe[7,8,9].

As adoption grows, so does the number of people needed to sustain the software. These roles must be funded, so each step in the lifecycle is associated with decisions on how to raise appropriate finances. Funding may be sought from a research funder or through commercial means. (It is noted that commercialisation is often difficult in an academic environment due to restrictions - e.g. open-source publication requirements by the original funder, the needs of an Open Science policy - and because academia rarely presents a user base of sufficient size to make commercialisation viable.) Alternatively, an open-development approach might be adopted where the software is provided for free and developers are attracted to work on the project by incentives (such as preferential support, or simply the kudos from being associated with a successful project).

At some point in its lifecycle, software may face an obstacle to its continued existence. It might fall from favour, be superseded, become redundant, run out of funding or simply lose a key developer. If the obstacle is insurmountable, the software can no longer be sustained and an approach to software preservation must be chosen.

Footnotes

6 "Software rot - Wikipedia, the free encyclopedia." 2011. 21 Dec. 2015 (**https://en.wikipedia.org/wiki/Software_rot**).

7 See **Appendix 1**, presentation 6.

8 "BALL - Biochemical Algorithms Library — The BALL Website." 2005. 27 Oct. 2015 (**ball-project.org**).

9 "figshare - credit for all your research." 2008. 27 Oct. 2015 (**http://figshare.com**).

3.5 Approaches to sustainability and preservation

In 2010, the UK's Software Sustainability Institute and Curtis+Cartwright investigated approaches to software preservation[10] which have since been adopted and developed by the scientific computing department at the Science and Technology Facilities Council in the UK.

1. **Encapsulation**. Preserve the original hardware and software to prevent change and ensure that the software continues to operate

2. **Emulation**. Emulate the original hardware and operating environment to give the appearance that nothing has changed so that the software continues to operate

3. **Migration**. Update the software to maintain the original functionality and transfer it to new platforms as necessary to prevent obsolescence

4. **Cultivation**. Keep the software up to date by adopting an open development model that allows new contributors to be brought on board

5. **Hibernation**. Preserve knowledge of how to resuscitate the software's exact functionality at a later date

6. **Deprecation**. Formally retire the software. Unlike hibernation, no time is invested into preparations to make it easier to resuscitate the software

7. **Procrastination**. Do nothing

Emulation software like CernVM[11], Docker[12], VMware[13], VirtualBox[14], and Vagrant[15] are seeing increasing adoption across the research community. These systems can be used to archive a copy of software and the environment in which it runs. They are proving useful for purposes of reproducibility, and are also being used to deliver and deploy software. A project in the UK, **recomputation.org**[16], is using virtual machines to store everything needed to replicate an experiment. The research software, data and the operating environment on which the computation occurred are stored together on a virtual machine. This is archived and can be accessed at a later date if the computation needs to be verified.

Footnotes

10 Hong, N. Chue, et al. "Software Preservation Benefits Framework." Software Sustainability Institute Technical Report (2010).
11 "CernVM." 2005. 28 Oct. 2015 (**http://cernvm.cern.ch**).
12 "Docker - Build, Ship, and Run Any App, Anywhere." 2013. 28 Oct. 2015 (**docker.com**).
13 "VMware Virtualization for Desktop & Server, Application ..." 28 Oct. 2015 (**vmware.com**).
14 Downloads – Oracle VM VirtualBox." 2011. 28 Oct. 2015 (**virtualbox.org/wiki/Downloads**).
15 "Vagrant." 2013. 28 Oct. 2015 (**vagrantup.com**).
16 See **Appendix 1**, presentation 7.

4 Key recommendations

There are some changes - some simple, others not - that could revolutionise software sustainability. Although many different views were presented at the workshop, there was agreement that the following recommendations would improve the reliability and reproducibility of research.

1. We must raise awareness of the fundamental role of software in research

- All stakeholders, from researchers to policymakers, must be included in this awareness raising campaign
- Incentives must be determined to persuade stakeholders to invest resources into software sustainability, and the risks of overlooking sustainability must be made clear

2. Research software should be recognised as a valuable research object in line with the investment it receives and the research it makes possible

- Research software should become a citable scientific deliverable of equivalent value to researchers as that of a publication
- Agreement should be made on methods for citing software
- Agreement should be made on methods for measuring research impact

3. Funders should use their position to promote software sustainability

- The goals of funders and software sustainability are well aligned
- Funding should be made available for maintaining, extending and preserving software
- Software management plans should be adopted to motivate researchers to consider their use of software and their plans for sustaining it

4. Skills related to software sustainability must be embedded in the research community

- More researchers should learn basic software engineering skills
- Doctoral training programmes should include an element of basic software engineering
- Career paths for expert software developers (Research Software Engineers) should be adopted

5. We must create organisations (centralised or distributed) to act as focal points for software sustainability expertise

- Researchers cannot be expected to acquire all the skills that software sustainability requires, they must have access to experts
- Each country should create these focal points in a way that suits their research community
- These organisations should work together to share knowledge and expertise
- This network may be enhanced by a European organisation for software sustainability

5 Benefits of software sustainability

Funding for research is limited. If we are to gain the investment that software sustainability needs, we must show that it will lead to an overall benefit for the research community. This is certainly the case. Software sustainability creates reliable, reproducible and reusable software which leads to:

1. Trusted research

- Results that are generated by reliable, well tested software can be trusted. Reproducible software makes it easier for others to repeat computations to verify results

2. Increased rate of discovery

- Reusable software is easy to adopt and extend. By building on existing software, researchers can invest more time into research and avoid wasting time by replicating software that already exists

3. Increased return on investment

- Software is very expensive to develop. Reusing software rather than replicating it has the potential to save a significant amount of resources, which could then be invested into new research

4. Research data remains readable and usable

- Data is meaningless without the software needed to read and interpret it. Software that continues to function allows continued access and use of research data, aiding reproducibility and helping extract the greatest return from the investment made into collecting the data

6 Societal barriers to software sustainability

Improving software sustainability will require changes to the accepted practices of the research community.

6.1 A lack of awareness of software's role in research

Possibly the greatest issue that currently limits software sustainability is simply one of awareness. The research community does not recognise the scale of its reliance on software or the issues that limit software sustainability.

The effects of overlooking software are felt across the research community. Researchers overlook the fundamental contribution that software makes to the reliability and reproducibility of their results. Publishers overlook the need to identify software as a vital part of the publication process. Funders overlook the need to make funding available for maintaining software, and overlook the growing need to secure software experts on research projects. Research organisations overlook the need to build software expertise in their staff. Policymakers overlook the importance of software to research, and they apply impact metrics that often penalise time invested into software development rather than reward it.

It is clear that a sustained campaign of awareness raising is required across the research community. This campaign should be led by proponents of software sustainability but must include all research stakeholders if it is to be successful. The campaign will rely on developing and promulgating a set of arguments to convince research stakeholders that software sustainability is of fundamental importance to research.

6.2 A lack of identification and citing of software

Reproducibility requires that the exact version of the software used to generate a result must be known. This is difficult, because software constantly evolves. A similar problem has been faced with data, where identifiers such as a Digital Object Identifier (DOI)[17] or Uniform Resource Name[18] are used to provide an actionable, interoperable and persistent link to a specific data set. The same system could be used to identify software.

Software repositories provide somewhere to store software and record its version history (i.e a record of each iteration of the software from when it was first conceived to the current version). Some of these repositories, notably Github[19] and Zenodo[20], allow DOIs to be associated with versions of the software. The DOI or URN makes it significantly easier to identify the specific version of the software used to generate a research result, and it makes it significantly easier for a reader to access that software.

When a researcher builds on someone else's research, it is good practice to cite the paper that describes the research. Highly cited papers are one way of identifying important research, and they are used to gain recognition and reward for the author of the paper.

Footnotes

17 "Digital Object Identifier System." 2015. 28 Oct. 2015 (doi.org).
18 "Uniform Resource Name - Wikipedia, the free encyclopedia." 2011. 14 Nov. 2015 (https://en.wikipedia.org/wiki/Uniform_Resource_Name).
19 "GitHub - Where software is built." 2008. 28 Oct. 2015 (https://github.com).
20 "Zenodo." 2012. 28 Oct. 2015 (http://zenodo.org).

When a researcher builds on someone else's software, they are unlikely to even mention it in the paper. Citation of software is necessary not just for reproducibility, but also to credit the software developer who made the research possible.

Software citations would provide a clear way of recognising the contribution of software developers and infrastructure providers to the research process. However, they would also allow researchers to gain credit for developing software, after all, most research software is developed by researchers.

Although the technology exists to cite software, there is not yet any general agreement across the community of the necessity to do so, nor is there significant pressure on publishers to mandate the citation of software. If we are to increase the citation of software, we must persuade the research community of its importance, agree on an acceptable way to cite software and then persuade policymakers, funders and publishers to reward - or even mandate - software citation. There is a growing number of journals where a researcher can publish their software[21] and at least one journal dedicated to describing research software with high reuse potential[22].

6.3 A lack of understanding of licensing and ownership

A significant restriction on the re-use of software is a lack of understanding about licensing[23]. If the owner of the software does not describe how others can use it (i.e. by licensing it), then people will - quite rightly - be cautious about using or extending the software.

The main problems with licensing relate to who owns the software and what licence best suits the owner's needs. The owner of the software is the only person who has the right to licence it. Before a licence can be chosen, the owner must be identified. Many research organisations do not make clear their stance on the ownership of software developed by people working within their organisation, yet the law is clear on the subject, so there is little reason for this position to persist. Licensing would become significantly easier if research organisations were to make a clear statement about their stance on the ownership of software developed by their employees (and other notable groups, such as students).

Raising awareness of licensing and providing easier access to Intellectual Property (IP) experts would help developers understand the correct licence to choose, and researchers to understand the limitations on their use of software. Many research organisations provide access to these experts but, anecdotally at least, it would appear that people are not making use of this service or that the experts are not sufficiently well versed in IP in relation to software.

6.4 A lack of clear incentives and impact

Software sustainability requires the investment of effort, changes in practice and funding. If we are to persuade stakeholders to make this investment, we must be clear of the incentives for doing so.

The worthiness of research is measured through impact metrics - generally related to publications and citations. These metrics add a strong incentive for researchers to be the first to discover and publish. This pressure to quickly gain results forces researchers to invest minimal time on any aspect of software development that is not focussed on generating a result - this is not an environment that fosters sustainability. We must devise research impact metrics that reward advances in research, but also reward making those advances using software that is sustainable.

As discussed above, if software were cited and made discoverable in publications, it would become an additional incentive for developing sustainable software.

6.5 A lack of software skills

Many researchers know how to code, but few understand the wider set of skills that are needed to develop reliable, reproducible and reusable software. These skills are generally understood as "software engineering".
Not all researchers have to become software engineers. Instead, it is necessary for a researcher to learn skills appropriate for their level of involvement in software. Organisations like Software Carpentry[24] address this problem by training researchers in the basics of software engineering. This type of training helps increase the general skill level of the research community, but if we are to significantly increase skills, software engineering should be incorporated into doctoral training programmes. Ensuring that software skills are provided at the very start of a research career is likely to ensure that these skills are used throughout that research career.

At some point during the growth of a software project, the skills and tasks that can be undertaken as a researcher become too great. At this stage it is necessary to acquire the skills of a software expert.

6.6 A lack of a career path for software experts

The goals of software sustainability would be considerably advanced if experts in software development and engineering were embedded in research groups. Handing over responsibility for software sustainability to an expert reduces the pressure on researchers to master yet another skill set.

Currently, software experts in academia lack a career path, so most are hired as postdoctoral researchers. Success in these positions is measured on the basis of publications, so a software developer hired into this role quickly finds themselves in a dead-end position with no option for recognition, reward or career advancement. This makes it difficult to hire and retain expert software developers in academia, and this lack of experts is a significant inhibitor of software sustainability.

Gaining recognition and reward for expert software developers is the goal of the UK's campaign for "Research Software Engineers" (RSEs), which is being led by the Software Sustainability Institute. Since 2013, the campaign has raised awareness of the RSE role, led to the founding of an association[25] to represent people in the role (with almost 500 members, 150 of whom joined in 2015), and has been successful in seeing funders and organisations accept the role - culminating in a £3 million investment in 2015 by the EPSRC to fund RSE Fellowships[26]. Within the UK, the term RSE is becoming the accepted way of referring to a person conducting software development in academia, and the use of the term is growing across Europe, Canada and the US.

Access to software experts will increase the sustainability of software developed within academia. But until there is a career path into which these experts can be hired, academia will continue to experience serious problems attracting and retaining software experts.

6.7 Gender balance

Gender is not an issue that directly affects software sustainability. However, there are significantly fewer women in software-related roles than men - a disparity that increases with seniority of role. Indeed, this gender disparity was mirrored by the attendees at the Knowledge Exchange Workshop on Research Software Sustainability. Although gender does not have a direct impact on software sustainability, it was generally agreed that diversity issues should be addressed where possible as part of this work.

Footnotes

21 "In which journals should I publish my software? | Software ..." 2013. 14 Nov. 2015 (**software.ac.uk/resources/guides/which-journals-should-i-publish-my-software**).

22 "Journal of Open Research Software." 2012. 14 Nov. 2015 (**http://openresearchsoftware.metajnl.com**)

23 See **Appendix 1**, presentation 4.

24 "Software Carpentry." 2014. 28 Oct. 2015 (**https://software-carpentry.org**).

25 "The people behind research software." 2006. 21 Dec. 2015 (**rse.ac.uk**).

26 "Research Software Engineer (RSE) Fellowships - EPSRC ..." 2015. 28 Oct. 2015 (**epsrc.ac.uk/funding/calls/rsefellowships**).

7 Technical barriers to sustainability

Some elements of software sustainability would be easier to achieve if certain technical barriers could be overcome.

7.1 Identifying good software

Adopting software requires a significant investment of resources. Researchers' resources are limited, so they can be reluctant to adopt software simply because it is too risky. This risk would be considerably reduced if there was a way of identifying **good** software. This would improve adoption and reuse of software - which are both goals of sustainability.

The needs of researchers vary widely so what is good is highly subjective. However, there is at least agreement on some elements that are important: that software is available, adequately documented, licensed, version controlled[27], tested, etc. If these elements could be understood and rated, it should be possible to create a system that will at least provide a measure of the software.

The subjectiveness of classifying software means that this is a highly controversial subject. However, it should be noted that similar concerns were raised about judging the quality of data, and there are efforts, such as the Data Seal of Approval[28] and the Five Stars of Open Data[29], that have succeeded in reaching widespread adoption. There is considerable interest in repeating this endeavour for software[30,31].

A framework proposed by the Netherlands eScience Center (NLeSC) looks to understand the requirements of both data stewardship and software sustainability[32] and combine them into a series of reusable protocols that define the usage of software and data in a specific research scenario.

The framework identifies three separate stakeholders:

1. Governments, research organisations and funding organisations
2. The research community, society, industry
3. Executive level parties, such as computing centres, data centres, libraries and policy organisations

Whenever a new software tool is used to generate research, the researchers who conduct the research also write a protocol. This describes the regulations and best practices that govern the software and data - like a combined software and data management plan - which includes aspects such as reusability, future access, and re-traceability of the data and software. The protocol is published at the same time as the paper that describes the research. In doing so, interested parties can read the paper to understand the research, then read the protocol to understand how the software and data were managed.

Since the protocol is now open, other researchers who intend to conduct similar research can reuse the protocol rather than having to write their own (or update or expand it, if necessary). This means the original authors of the protocol are rewarded with a citation for investing the time into writing the protocol. In other words, the protocol will become a parallel means for authors to gain credit alongside journal publications.

It is the responsibility of the category 1 stakeholders (governments, research organisations and funding organisation) to set up the framework, determine minimum requirements for its use, provide guidelines on how to use it, and link to relevant laws. It is the responsibility of the category 2 stakeholders (the research community,

society and industry) to set up expert groups within a discipline and make them responsible for the set up of protocols, to write the protocols whilst adhering to the guidance of the general framework, and then publish the protocols as scientific publications.

A similar effort is underway in the UK, led by the Software Sustainability Institute, to create a "Software Accreditation Framework" which could be the basis of the Software Seal of Approval. This would be a lightweight framework that allows researchers to understand their software sustainability practices by assessing their work against a checklist of good practice. Details should be publicly available in 2016.

7.2 Software discovery

Even **good** software will not be used if researchers are not aware of its existence. Making software more discoverable should reduce replication and foster reuse. More reuse will allow funding and researcher effort to be focussed on research rather than software development. If developers can show use of software, they will find it easier to make a case for funding to maintain their software.

However, making software available is more complicated than simply adding a download to a website (referred to by the developer's adage "Build it and they will come"). Research software tends to have a highly specific purpose, so much of it has a very limited potential market. Making people aware of software requires skills in marketing and advertising that are often lacking in research projects. Software catalogues and brokers are two approaches to increase awareness and availability of research software.

Software catalogues are typically created to benefit a particular field or a particular set of computational processes. Since they are made to service a particular market, they act as a single point of contact around which a community may grow, which increases the likelihood that software will be discovered. Software catalogues come in two forms. The first is most like a catalogue: a description of the different software available and links to the developers of that software. The second is more centralised: a software repository (separate from the developers' own storage infrastructure) where software is stored.

The main problem with software catalogues, especially those that remove the software from the original developers, is that they require significant curation effort. If the catalogue is not reviewed and refreshed on a regular basis, it risks becoming a collection of out of date software - a **software graveyard** rather than a software catalogue.

Software brokers[33] are an active form of catalogue that tries to match a researcher's needs with a particular technology. Software developers would use the broker to describe their software's functionality and researchers would use it to describe their requirements. The broker could then match researchers and software developers.

There is an obvious problem that the software catalogues and brokers can only operate if they have access to information about the software that is available. One solution that fits with other proposals in this report is to mine Software Management Plans for this information.

Footnotes

27 See **Appendix 1**, presentation 5.
28 "Data Seal of Approval: Home." 2009. 14 Nov. 2015 (**http://datasealofapproval.org**).
29 "5-star Open Data." 2012. 14 Nov. 2015 (**http://5stardata.info**).
30 "The five stars of research software | Software Sustainability ..." 2013. 14 Nov. 2015 (**software.ac.uk/blog/2013-04-09-five-stars-research-software**).
31 See **Appendix 1**, presentation 10.
32 See **Appendix 1**, presentation 2.
33 See **Appendix 1**, presentations 1 and 8.

8 Providing access to expertise in software sustainability

If we are to improve the research community's use of software, there must be an organisation (or organisations) who promote software sustainability and provide access to expertise in the subject.

Software sustainability is a broad issue, so these organisations would have to be equally diverse because they might be called upon to provide researchers with expertise on everything from software engineering to community building.

8.1 Centralised expertise: the UK's Software Sustainability Institute

The UK was the first country to invest in an organisation that focusses on improving software sustainability. The Software Sustainability Institute is a national service for UK-based researchers. It's first phase (2010-2015) was funded by the Engineering and Physical Sciences Research Council. Its second phase (2015-2019) has attracted two additional funders: the Biotechnology and Biological Sciences Research Council and the Economic and Social Research Council. These additional funders is testament to the growing interest in software sustainability across disciplines.

Software sustainability is a broad issue, which the Institute tackles by splitting its effort over five teams.

- **The Community team** gathers information from the research community. It runs workshops and a fellowship programme to raise awareness of software sustainability and gather information about software from the research community
- **The Research Software team** comprises research software engineers who provide their expertise in software engineering to the research community[34]
- **The Training team** facilitates and runs training, such as Software and Data Carpentry, to provide the skills needed by the research community
- **The Policy team** conducts research to understand the research software community and the investment into it, then uses this information to campaign for changes to be made that support sustainability
- **The Communications team** have developed a highly popular platform which is used to encourage participation from the research community and to disseminate information about software sustainability

The Institute has successfully raised awareness of the issues around software sustainability, and created a significant community around those issues[35]. Endeavours like the Research Software Engineer campaign[36] are changing the research community, and the Journal of Open Research Software[37] is changing the way researchers think about software publication.

8.2 Distributed expertise: DANS and SURFsara

A centralised approach to software sustainability arose in the UK due to a confluence of expertise and funding. It is not necessarily the only way to approach sustainability. Sharing expertise across a number of centres is also a possibility and is the approach that is being considered in the Netherlands with both DANS[38] and SURFsara[39] sharing responsibility.

To work effectively, there should be a clear demarcation of responsibilities between each centre, and good communication, so that lessons learned through dealing with projects can propagate across the centres.

8.3 Building capacity across Europe

There is obviously a need for sustainability expertise to be housed within each country. We benefit from a highly active and well-funded research infrastructure across the EU, which raises the question of whether some broader Europe-wide organisation should also be formed. This would most likely act as an umbrella group that shares expertise across national organisations and campaigns for software sustainability at an EU level.

There was broad agreement at the workshop that an EU-level organisation or platform should be investigated. This investigation should include representatives from the different European countries that have expressed an interest in a centre for European software sustainability.

It was noted at the workshop that the concept of Europe-wide software sustainability could be raised at the next meeting of the Research Data Alliance[40], since many of the people involved in this project will have an interest in software sustainability.

Footnotes

34 See Appendix 1, presentations 3.
35 "Measuring the success of the Software Sustainability Institute." 2015. 28 Oct. 2015 (software.ac.uk/attach/SuccessMetricsApril2014.pdf).
36 "Research Software Engineers | Software Sustainability ..." 2015. 14 Nov. 2015 (software.ac.uk/policy/research-software-engineers).
37 "Journal of Open Research Software." 2012. 14 Nov. 2015 (http://openresearchsoftware.metajnl.com).
38 "DANS — English." 2005. 14 Nov. 2015 (dans.knaw.nl/en).
39 "SURF | SURFsara." 2015. 14 Nov. 2015 (surf.nl/en/about-surf/subsidiaries/SURFsara).
40 "RDA | Research Data Sharing without barriers." 2013. 21 Dec. 2015 (https://rd-alliance.org).

9 The role of funders in software sustainability

Funders look to support reliable, trusted research which is also the goal of software sustainability. A lack of awareness of the importance of software is a significant barrier to software sustainability, and funders are well placed to change this situation.

They can raise awareness of software sustainability, they can ask researchers to describe their plans for software-related issues in their proposals, and they can incentivise this change by making funding contingent on adequate software planning.

9.1 A change to novelty requirements

Funders seek novelty in the proposals they fund, but this is antithetical to the needs of software sustainability, where maintenance - not novelty - is key. This focus on novelty provides a perverse incentive for researchers to develop new software rather than reuse existing software. Funders could promote reuse of software by ensuring that reviewers of proposals look for novelty only in research and not in the software used in that research.

9.2 Funding for maintenance, not just creation

If software is not to succumb to decay, it must be maintained. Before software can be reused, it typically must adapted or extended to its new purpose. Maintaining and extending software are core requirements of software sustainability, but very little funding is available for these purposes.

The difficulty of gaining funding for software maintenance is a significant barrier to the sustainability of software. If funders are to foster sustainability, they must provide funding for maintaining, adapting and extending software. An alternative approach would be to create a separate funding stream purely for investment into software maintenance projects. The UK's Engineering and Physical Sciences Research Council and the Biotechnology and Biological Sciences Research Council have both run funding calls for the maintenance of software. These schemes - **Software for the Future**[41] and **Bioinformatics and Biological Resources Fund**[42] - have been highly competitive, proving significant demand for such schemes.

9.3 Software Management Plans

Data Management Plans (DMPs)[43,44] have been mandated by many funders. This has forced researchers to contemplate the importance of their data, how it should be identified and preserved, and how value can be best extracted from it. The mandate for DMPs has cost little - in both the effort needed to implement the change and in the extra effort required to complete the plan - but has led to a significant improvement in awareness of data issues. Similar benefits could be gained if funders were to mandate Software Management Plans (SMPs)[45].

It must be noted that aside from their passing resemblance, DMPs and SMPs will require significantly different information and will require significantly different expertise to complete. This means that it is inadvisable to simply extend DMPs to cover software. To improve software sustainability, SMPs must be viewed as separate and different to DMPs.

Software Management Plans (SMPs) chart the development and sustainability plans for software used and developed in a research project. They force researchers and developers to contemplate their software's lifespan and to make plans for sustaining it in the future.

A typical SMP will require information about the software assets (i.e. what software will be used and developed), intellectual property and governance, provisions for access, sharing and reuse, plans for long-term preservation, and resourcing and responsibility. A prototype service to help developers to write an SMP has recently been released by the Software Sustainability Institute, which could be adapted and extended for use by funders (it is currently undergoing an upgrade and as such will not be available until 2016).

Some research funders have made SMPs a mandatory aspect of specific funding calls related to software development. If this practice were to grow, for example if SMPs were to be a mandatory part of any research project in which software was developed, then recognition of software sustainability would increase significantly. Assessing SMPs will require funders to acquire people with the necessary skills, potentially through including technical peer reviewers, such as research software engineers (see above), on proposal review committees.

SMPs could also help foster reuse of software if they were to require that anyone seeking funding for software development must first show that reasonable effort has been invested into checking that the software does not already exist. If similar software is found, the developers would be required to explain why reuse is not possible and hence why new development is needed. The information in these SMPs could be extracted to provide an overview of the research software that is available, which would be a valuable resource for improving awareness of software across the research community.

It should be noted that SMPs should not become another part of the paperwork that researchers must fill in for no obvious benefit. To ensure they are useful, and do not stifle innovation, SMPs should be lightweight and necessary only when software is to be developed by a research project.

SMPs will only affect change if they are assessed as a vital part of a funding proposal, and if they are accepted by the research community: a good SMP should increase the proposal's chance of being accepted.

Footnotes

41 "Software for the Future II - EPSRC website." 2015. 13 Nov. 2015 (epsrc.ac.uk/funding/calls/softwarefuture).

42 "Bioinformatics and Biological Resources Fund - BBSRC." 2015. 13 Nov. 2015 (bbsrc.ac.uk/funding/opportunities/2014/2014-bioinformatics-biological-resources-fund).

43 "Data Management Plan (DMP) | Data - Research Data ..." 2014. 21 Dec. 2015 (https://data.uni-bielefeld.de/en/data-management-plan).

44 "DMPonline." 2011. 21 Dec. 2015 (https://dmponline.dcc.ac.uk).

45 "Writing and using a software management plan | Software ..." 2014. 21 Dec. 2015 (software.ac.uk/resources/guides/software-management-plans).

10 Current national activities[46]

10.1 Germany

Presented by Timo Borst, head of innovative information systems and publishing technologies, ZBW, German National Library of Economics[47].

The view from Germany on software sustainability was presented from the perspective of infrastructure providers. Maintenance of software is a problem because, although many researchers develop their own software, they lack experience in ensuring software quality or ease of reuse, and often lack extra information needed to make use of software, such as documentation and licensing information. The role of software maintenance cannot be taken on board by infrastructure providers, because they lack the resources to do so. What's more, resources are difficult to come by because funders tend not to invest in maintenance, being as they are supporters of novelty, not duration.

A distinction was drawn between **research software**, which is developed by researchers, and **infrastructure software**, which is developed by interdisciplinary teams for infrastructure purposes. These different types of software are developed under different circumstances with different target audiences. **Research software**, under this definition, is developed to achieve a research goal with no effort invested into maintenance (i.e. on reuse or integration with infrastructure). However, **Infrastructure software** is developed with a focus on creating a data structure or on data management and, as a consequence, some maintenance is planned.

Infrastructure providers can help support software maintenance and foster its reuse. They can help develop and disseminate best practice, provide platforms for developing software (via PaaS and IaaS) and provide a long-term archiving service.

To combat problems with software citation, the success of the DataCite project was discussed. This allowed data to be associated with a DOI, which allowed data to be cited in papers to the benefit of both reproducibility and the recognition of the role of the data. There is interest in providing a similar service for software.

There have been some projects, workshops and initiatives to foster software sustainability.

It was noted that the DFG only provides funding for software development and not software maintenance. Funders must fund software maintenance if we are to expect stakeholders to conduct software maintenance.

10.2 The Netherlands

Presented by Peter Doorn, director, Data Archiving and Networked Services[48].

Interest in software sustainability has grown in the Netherlands over the last three years. Within the humanities, concerns have been raised by digital humanities programs and infrastructures, such as CATCH[49] and CLARIN[50]. These were mirrored by rdnl[51] (a cooperative comprising DANS[52], 3TU.Datacentre[53] and SURFsara[54]) which was created to investigate long-term data archiving, but developed an interest in software when it was realised that this was fundamental to their goals.

A series of meetings has been held since 2012, which ultimately led to the creation of a report **Self Service for Software Sustainability**, which has not been released publicly, but a summarised version will soon be made more widely available. The summarised version, called **Research Software at the Heart of Discovery** covers some major issues with software sustainability, such as how best to utilise existing organisations and models to develop a federated/virtual Dutch Software Sustainability Institute, developing software- and data-carpentry training for new PhDs, promoting research software as a citable scientific deliverable of equivalent value to publication, ensuring recognition and career paths for Research Software Engineers and developing academic models for sharing and disseminating research software.

One important aspect of the report is a discussion of a **seal of approval** for software, as described above. This approach is based on the successful DataSHIELD project, launched in 2005, which provided researchers with a framework against which a minimum quality standard for data could be developed. A similar initiative is hoped to be successful for software.

The next step in the Netherlands investigation of software sustainability is to release the **Research Software at the Heart of Discovery** report to important stakeholders and collect responses. This will be performed whilst investigating the viability of the software seal of approval. A second step is to investigate the viability of a European Software Sustainability Institute. This would be an organisation that borrows from its UK counterpart and provides some level of services to researchers across Europe.

10.3 The United Kingdom

Presented by Matthew Dovey, Head of research technology, Jisc[55].

The Software Sustainability Institute was founded in the UK in 2010 to investigate and tackle the issue of software sustainability. It has conducted much of the early work and developed approaches to foster sustainability in a way that meets the requirements of researchers. See above for a detailed description of the Institute.

Collaborative Computational Projects (CCPs) bring together leading UK expertise in key fields of computational research to tackle large-scale scientific software development, maintenance and distribution. Each project represents many years of intellectual and financial investment. The aim is to capitalise on this investment by encouraging widespread and long term use of the software, and by fostering new initiatives such as High End Computing consortia.

The CCPs are supported by the Software Engineering Support Centre (SESC), which runs a community hub for software, and provides access to software development tools and specialised advice. The goal of the SESC is to promote sustainability of the CCPs by supporting software projects to remain usable - even after active development ends. The SESC also provides outreach around software credit and work on embedding automated testing.

Footnotes

46 There is no information from Denmark or Finland because representatives from these countries did not attend the workshop.
47 See **Appendix 1**, presentation 11.
48 See **Appendix 1**, presentation 10.
49 "Continuous Access To Cultural Heritage (CATCH) - NWO." 2013. 21 Oct. 2015 (**nwo.nl/en/research-and-results/programmes/Continuous+Access+To+Cultural+Heritage+(CATCH)**).
50 "CLARIN: Common Language Resources and Technology ..." 2015. 21 Oct. 2015 (**https://eudat.eu/communities/clarin-common-language-resources-and-technology-infrastructure**).
51 "Research Data Netherlands." 2013. 21 Oct. 2015 (**researchdata.nl**).
52 (**dans.knaw.nl**).
53 "Search - 3TU.Datacentrum." 2010. 21 Oct. 2015 (**http://datacentrum.3tu.nl/en/home**).
54 "SURF | SURFsara." 2015. 21 Oct. 2015 (**surf.nl/en/about-surf/subsidiaries/SURFsara**).
55 See **Appendix 1**, presentation 9.

Jisc is investigating a number of areas related to software sustainability and data. The Research Data Spring (RDS)[56] aims to find new technical tools, software and service solutions that will improve researchers' workflows and the use and management of their data. It is a partnerships between researchers, librarians, publishers, developers and other stakeholders engaged in the software and data lifecycle. It involves a series of sandpits where researchers pitch ideas, some of which are funded by Jisc. One of these sandpits led to the **Software reuse, repurposing and reproducibility project**[57] which aims to challenge some of the major software sustainability challenges, such as discovering and reusing software.

10.4 Outside Europe

Efforts to improve software sustainability are not limited to Europe.

In Australia, the Research Platform Services[58] based at the University of Melbourne has taken the lead in providing researchers with the research-specific software. Their main contribution to software sustainability has been in promoting the adoption of Software Carpentry across departments at the University of Melbourne and at other universities in Australia. They also run a training programme, called Research Bazaar[59], that has a strong focus on community building and the adoption of software engineering and data management techniques.

In Canada, CANARIE's[60] Research Software Program[61] champions the development of software tools that accelerate discovery by simplifying access to digital infrastructure. A series of workshops and funding calls have raised awareness of research software, have supported careers for software experts in academia and has led to the development and support of new software tools.

The main endeavour in the US is the Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE)[62]. Although it started as a forum for discussion of the challenges faced by software sustainability, it now encompasses a significant number of people who are lobbying for changes to software practice in the US and who are looking to create centres to support software sustainability.

Depsy[63] promotes credit for software as a fundamental building block of science. It text-mines papers to find fulltext mentions of software they use, revealing impacts invisible to citation indexes, like Google Scholar.

Footnotes

56 "Research data spring | Jisc." 2015. 14 Nov. 2015 (**jisc.ac.uk/rd/projects/research-data-spring**).
57 "Research data spring: software reuse, repurposing ... - Jisc." 2015. 14 Nov. 2015 (**jisc.ac.uk/podcasts/research-data-springsoftware-reuse-repurposing-and-reproducibility-22-sep-2015**).
58 "About us - The Research Bazaar - ResBaz." 2015. 16 Nov. 2015 (**http://melbourne.resbaz.edu.au/about**).
59 "The Research Bazaar." 2015. 16 Nov. 2015 (**http://melbourne.resbaz.edu.au**).
60 "CANARIE Network | CANARIE." 2014. 14 Nov. 2015 (**canarie.ca/network**).
61 "Research Software | CANARIE." 2014. 14 Nov. 2015 (**canarie.ca/software**).
62 "Working towards Sustainable Software for Science: Practice ..." 2013. 14 Nov. 2015 (**http://wssspe.researchcomputing.org.uk**).
62 DEPSY project (**http://depsy.org**).

11 Appendices

Appendix 1: presentations from the workshop and key message

No.	Title	Presenter	Key message
1	Transition – Software Sustainability Explained	Daan Broeder- Meertens Institute, Netherlands	When considering software sustainability try to cater for developers working within or close to the research groups. Provide them with a long-term perspective and career opportunities. In the end, the most sustainable software is the most used software.
2	Responsibilities - Data Stewardship and Software Sustainability	Patrick Aerts - the Netherlands eScience Center (NLeSC)	It is essential that researchers themselves address the issues of data stewardship and software sustainability. But they need support and guidance, based on minimum requirements, best practices, legal matters and standards. Software sustainability and data stewardship should be addressed on an equal footing. A European Software Sustainability Initiative supported by all countries should be considered.
3	Transition: Towards reusable and extensible code, SSI's Open Call approach	Neil Chue Hong - Software Sustainability Institute, UK	This presentation provided general information about the topic addressed, indicating the importance to raise awareness and to acknowledge the variety of stakeholders and necessity of coordinated approaches.
4	Re-use: Stimulate sustainability with strong compatible licenses and additional license provisions	Martin Hammitzsch - Research Centre for Geosciences, Germany	To achieve specific goals when releasing software consider licence up- and down-stream compatibility, select a strong permissive or copyleft licence, and provide additional licence provisions. In the sciences a toolbox covering these aspects and a guideline may help to make the best decision.
5	Development – An HPC service provider and industry client, merging software using Git	Cedric Nugteren - SURFsara, Netherlands	Collaboration between researchers, industry and HPC-centres can be facilitated using a versioning system such as git as long as a clear flow is agreed upon. Sustainability can be further improved using formalised tests, checks, and (in-line) documentation.
7	Long Term Maintenance - BALL - Biochemical Algorithms Library, keep software depending on operating systems or underlying software up-to-date	Andreas Zeller - Saarland University, Germany	This presentation provided general information about the topic addressed, indicating the importance to raise awareness and to acknowledge the variety of stakeholders and necessity of coordinated approaches.

No.	Title	Presenter	Key message
8	Research Perspective	Hans Bennis – Meertens Institute, NL	It is essential that thinking about software sustainability becomes a natural part in the development of research proposals that include the creation of software. This can be achieved by requiring researchers to pay attention to software sustainability in their proposals as part of the assessment criteria in the review procedure (as is often the case for data) and by developing procedures for the assessment of quality of software in a kind of Software Seal of Approval.
9	National situation UK	Matthew Dovey - Jisc, UK	This presentation provided general information about the topic addressed, indicating the importance to raise awareness and to acknowledge the variety of stakeholders and necessity of coordinated approaches.
10	National situation Netherlands	Peter Doorn - DANS, Netherlands	This presentation provided general information about the topic addressed, indicating the importance to raise awareness and to acknowledge the variety of stakeholders and necessity of coordinated approaches.
11	National situation Germany	Timo Borst - ZBW, Germany	To become more sustainable and integrated, research software must comply with the standards of a more general research infrastructure. Infrastructure providers, on the other hand, can contribute by maintaining, updating and preserving those pieces of software, which turned out to be important for the research community.

Appendix 2: position papers and statements from attendees at the workshop

- “Relevance and Challenges regarding Research Software Sustainability”, Hammitzsch, Martin; Wächter, Joachim. (**http://dx.doi.org/10.5281/zenodo.35360**)

Knowledge Exchange Office
C/ O Jisc,
One Castlepark,
Tower Hill,
Bristol, BS2 0JA

t: +44 203 697 5804
e: office@knowledge-exchange.info